Efficient nondestructive equality checking for trees and graphs

Michael D. Adams  R. Kent Dybvig

ICFP, 2008
Overview

1. Background
2. Basic Implementations
   - R5RS Implementation
   - Union-find implementation
3. Optimized Implementations
   - Precheck
   - Interleaved
   - Interleaved with Precheck
4. Summary
Background

Revised\textsuperscript{5} Report on the Algorithmic Language Scheme (R5RS):

- “Equal? may fail to terminate if its arguments are circular data structures”
Scheme Request for Implementation 85 (SRFI-85):

- Defines `equiv?`
- Terminates on all input
- Reference implementation
Background

Revised\textsuperscript{6} Report on the Algorithmic Language Scheme (R6RS):

```
• “The equal? predicate returns #t if and only if the (possibly infinite) unfoldings of its arguments into regular trees are equal as ordered trees.”
```
Requirements

- Correct
- Unknown inputs
- Not slow
R5RS (Tree) Equality
(define (equal? x y)
  (cond
    [(eq? x y) #t]
    [(pair? x) (and (pair? y)
                       (equal? (car x) (car y))
                       (equal? (cdr x) (cdr y)))]
    [(vector? x) (and (vector? y)
                       ...)]
    ...)))
On non-sharing:
- Fast
R5RS (Tree) Equality

- On non-sharing:
  - Fast

- On cycles:
  - Loops forever
R5RS (Tree) Equality

- On non-sharing:
  - Fast

- On cycles:
  - Loops forever

- On acyclic sharing:
  - Exponential

Adams (Indiana Univ.) Nondestructive equality ICFP 2008
Union-Find (Graph) Equality
Union-Find: Algorithm

- Equivalence classes
- Each call
  - Trivial
  - Same equivalence class
  - Different equivalence classes
Union-Find
Union-Find: Analysis

- Union-find
  - Inverse Ackermann (almost constant)
- Equality
  - Almost linear
- DFA equivalence algorithms
Union-Find: Implementation

- Pointer per object
- Store in object header
  - Not pure/functional
  - Overhead to every object
  - Threads
Union-Find: Implementation

- Pointer per object
- Store in object header
  - Not pure/functional
  - Overhead to every object
  - Threads

- Use “eq” hash tables
  - Map object to pointer
  - GC Interaction
25x slower than tree-equality
- Hash table
- Union-find algorithm
Precheck
Algorithm

- Tree-equality for N steps
- After N steps, restart and run union-find

Basic idea in SRFI-85
Setting Precheck Bound

- Bound too big
- Bound too small
- No perfect bound
Interleaved Equality
Interleaved

- Want
  - Fast tree-equal on trees
  - Required union-find-equal on cycles/DAGS

- Idea
  - Co-routines
Interleaved

• Tree-equality for $k_0$ steps
• Union-find for $k_b$ steps
  • But continue if successful
Interleaved

- Tree-equality for $k_0$ steps
- Union-find for $k_b$ steps
  - But continue if successful

Synergy

- Tree parts get tree-equality
- Cycle and graph parts get union-find
Analytic Performance

- Within constant of union-find
- Start with union-find
  - $k_0/k_b$
- Start with tree mode
  - $k_0$
Where do we start?

- **Union-find**
  - Pro: $k_0/k_b$, large cycles
  - Con: small trees

- **Tree**
  - Pro: small trees
  - Con: $k_0$, large cycles
Where do we start?

- **Union-find**
  - **Pro:** $k_0/k_b$, large cycles
  - **Con:** small trees

- **Tree**
  - **Pro:** small trees
  - **Con:** $k_0$, large cycles

Neither solution acceptable
Interleaved with Precheck
Interleave and precheck

- Small trees
- Large cycles
Benchmarks

- Lists
- Rev. lists
- Rand. DAGS
- Balanced binary tree
- Degenerate DAGS
- Random graph
- Union-exercising graph
Benchmark

A dams (Indiana Univ.)

Nondestructive equality

ICFP 2008 27 / 30
Conclusions

- Simple idea, subtle performance implications
- Mixing best of all worlds
  - Tree
  - Union-find
  - Interleaving
  - Precheck
Conclusions

- Interleaving with precheck
  - Never the best
  - Always almost the best
- Particular applications vs. General purpose library
Questions?